Wacky Racers 2020 Instructions

Michael Hayes, Ben Mitchell

Version 2, February 11, 2020

1 Introduction

The purpose of this assignment is to design, build, and program an embedded system using an ARM microcontroller and surface mount technology.

The goal for each group of four students is to build a remote controlled vehicle (the Wacky Racer) and its controller (the Wacky Hat). At the conclusion of the assignment there will be a dastardly race!

Each group is comprised of two sub-groups of two students. One of these subgroups constructs the Wacky Racer and the other constructs the Wacky Hat. You may be asking why is the Wacky Hat called the Wacky Hat? Well, a hat that controls a remote vehicle using head motions is not an ordinary hat!

2 Requirements

The following requirements are mandatory if you wish to maximise your marks.

2.1 Wacky racer

1. The chassis is to be constructed by each group. These can be 3-D printed, constructed from Perspex or wood, etc. A standard chassis is available from the Automation Lab technician (Daniel Hopkins). The electronics must be visible on top of the chassis.
2. Have a standard working bump sensor (supplied).
3. Locomotion can only use two 6 V DC motors (supplied).
4. Everything must be powered from a single 6-cell NiMH battery pack (supplied).
5. Use a single four layer printed circuit board of dimension 85 mm×64 mm.
6. Use an ARM microcontroller (Atmel SAM4S8).
7. Drive the motors using H-bridges (Texas Instruments DRV8833 dual H-bridge is recommended).
8. Regulate the nominal battery voltage to 5 V with a buck regulator IC (ADP2302ARDZ-50).
9. Be decorated with an LED tape (supplied) controlled by the MCU.
10. Use a USB interface for debugging.

11. Use a serial wire debug interface for MCU programming/debugging.
12. Have adequate battery fusing and reverse polarity protection.
13. Have a sleep button.
14. If the battery voltage drops below 1 V/cell, an LED should flash and high power draw devices should be disabled.
15. Interface to the Wacky Hat with a Nordic nRF24 SMD radio module.
16. Have four jumper selectable radio channels.
17. Have a mounted RFID card (supplied) that can be read by the Wacky ramps.
18. Be humorous.

Each Wacky Racer can have a dastardly means of hindering another team's Wacky Racer. However, you cannot:

- Damage or destroy another Wacky racer
- Damage the venue
- Injure a spectator
- Jam the radio signals

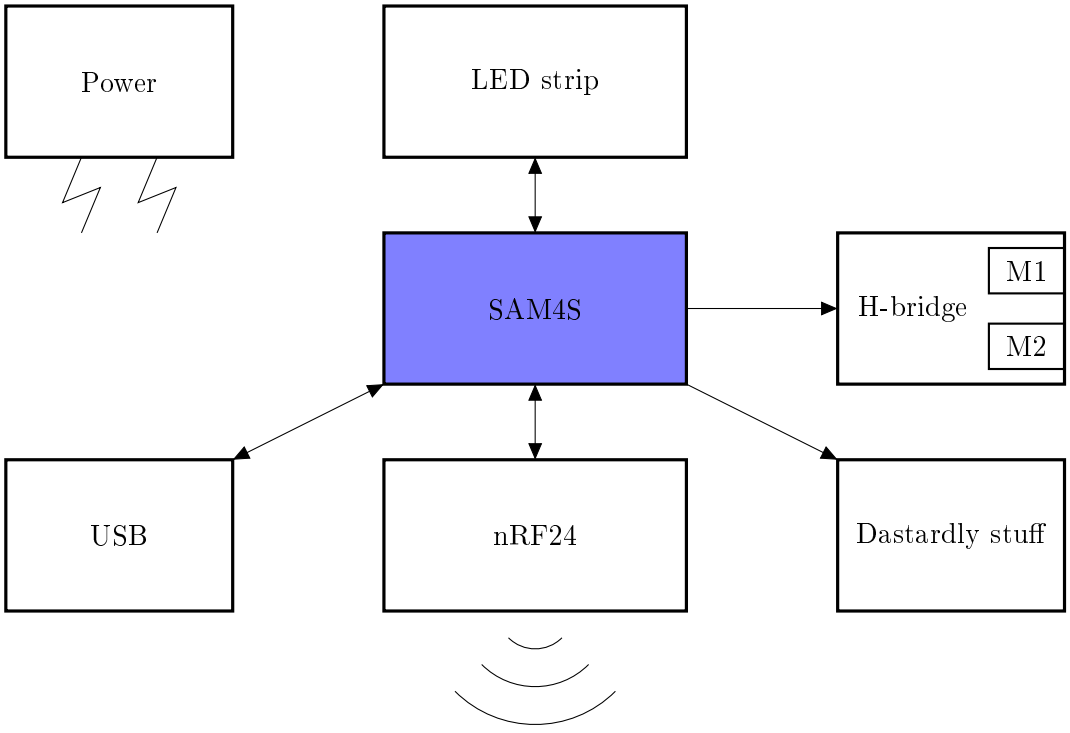

Figure 1: Racer board top level diagram.

2.2 Wacky hat

1. Construct a Wacky Hat that contains all the electronics.
2. Everything must be powered from a single 6-cell NiMH battery pack (supplied).
3. Have adequate battery fusing and reverse polarity protection.
4. Use a single four layer printed circuit board of dimension 85 mm×64 mm.
5. Use an ARM microcontroller (Atmel SAM4S8).
6. Regulate the nominal 6 V battery voltage to 5 V with a buck regulator IC (ADP2302ARDZ-50).
7. Be decorated with an LED tape (supplied) controlled by the MCU.
8. Use an I2C IMU (MPU-9250) for head motion detection.
9. Use a USB interface for debugging.
10. Use a serial wire debug interface for MCU programming/debugging.
11. Have a joystick in case the IMU does not work.
12. Have a sleep button.
13. If the battery voltage drops below 1 V/cell, an LED should flash and high power draw devices should be disabled.
14. Play sound when the bumper is hit.
15. Interface to the Wacky Racer with a Nordic nRF24 SMD radio module.
16. Have four jumper selectable radio channels.
17. Be humorous.

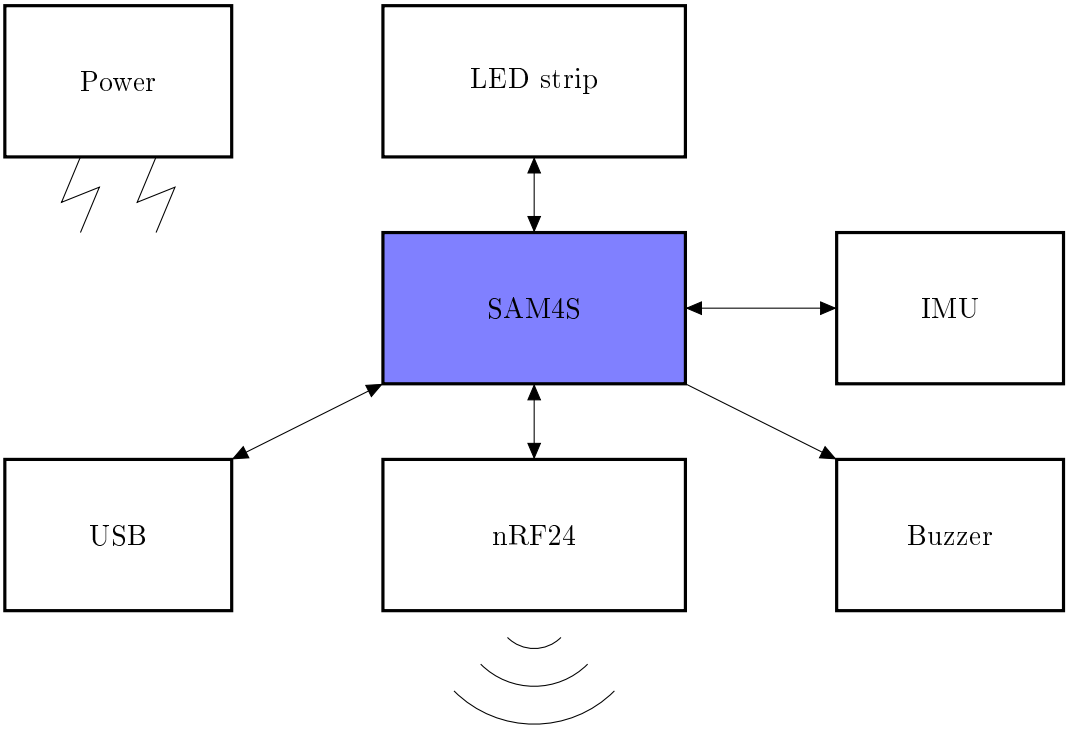

Figure 2: Racer hat top level diagram.

3 Assignment schedule

There is a planned activity for the timetabled labs in the Embedded Systems Lab (ESL):

Week	Assistance	Assessment
T1-1	Altium tutorial 1 (schematics)	
T1-2	Altium help	
T1-3	Schematic review	Schematic submission for review (Mon. noon)
T1-4	Altium tutorial 2 (PCB)	
T1-5	Altium help	PCB submission 1
T1-6	Altium help	PCB submission 2
T1-7	General	PCB submission 3
B-1	(break)	
B-2	(break)	
B-3	(break)	
T2-1	General	Blinky
T2-2	General	IMU/motor s
T2-3	General	Radio control
T2-4	General	Functionality
T2-5	Competition	Competition, critique

Notes:

1. There there may be a 6–10 day delay for the PCBs to be manufactured from the time of submission. You will then need to book an assembly session in the SMT lab.
2. Do not underestimate the blinky milestone. It requires having a functional PCB with a microcontroller that turns on properly, a functional toolchain and the ability to download code into the microcontroller's flash memory.

4 Assessment

The marks breakdown (max. 0x64) is:

PCB submission	0–0xa marks
Blinky milestone	0x5 marks
IMU/motor milestone	0x5 marks
Radio control milestone	0x5 marks
Functional assessment	0–0x4 marks
Board inspection	0–0x1e marks
Competition	0–0xa marks
Individual critique	0–0xf marks

4.1 Milestones

There are five milestones. To achieve the associated marks, they must be demonstrated to a T.A. by the end of the lab for your allotted stream. If you need an exception to this, see Ben Mitchell with a *very* good reason. The milestone requirements are:

Schematic review Submit your A3 schematic on Learn for review. Lose 10 marks if you miss the submission time.

PCB submission Submit your PCB design to Learn for manufacture. To encourage early submission there is a sliding scale for marks depending on when the PCB is submitted, see table. After week 7 there is a 10 mark penalty per week. **NB, a rushed PCB design will cause you more grief, more PCB rework, and a lower mark for the inspection.**

Week	Submission day	Cut-off time	Mark
5	Monday	10.00 am	10
5	Wednesday	10.00 am	9
5	Friday	10.00 am	8
6	Monday	10.00 am	7
6	Wednesday	10.00 am	6
6	Friday	10.00 am	5
7	Monday	10.00 am	4
7	Wednesday	10.00 am	2
7	Friday	10.00 am	0

Blinky Demonstrate that can blink an LED controlled by the SAM4S.

IMU/motors For the Wacky Hat, demonstrate output of IMU readings to a PC using USB CDC. For the Wacky Racer, demonstrate control of the motors from a PC using USB CDC.

Radio control Demonstrate sending commands from the Wacky Hat to the Wacky Racer over a radio link.

If you cannot show the functionality of a previous milestone during any assessment, you will fail that assessment and loose any marks from the previous milestone.

4.2 Functionality assessment

Functionality requirements:

Wacky racer	Wacky hat
Blink LED	Blink LED
Drive motors forward/backward	Read from IMU
Speed control of motors	Calculate speeds from IMU
Steering control	Joystick control
Receive radio message	Send radio message
Jumper selectable radio channel	Jumper selectable radio channel
Dies on bump	Plays sound on bump
Report group info to game server	Report group info to game server
Low voltage indication	Low voltage indication

Marks are allocated on how well things work. Up to 5 bonus marks can be awarded for extra functionality such as:

Wacky racer	Wacky hat
Dastardly stuff	Plays interesting sounds
Sleep mode	Sleep mode

4.3 Competition

The competition is a race around an obstacle course. Marks will be awarded every time you pass over a Wacky Ramp in the correct order. These ramps have an RFID card reader that will detect your Racer if you have correctly fitted an RFID card. The Wacky Ramps change colour when they detect a Wacky Racer.

To be awarded any marks for the race:

1. Your vehicle must stop for at least 5 s if the bumper is hit.
2. Your vehicle must be controlled by motions of the Wacky hat sitting on someone's head.

4.4 PCB inspection

This is assessed after the competition. The categories are:

1. Layout (component placement etc.)
2. Construction (tidiness, rework, etc.)
3. Testability
4. Power supplies (routing, decoupling, etc.)

5 Technical stuff

Read this section carefully. There are clues as to how we mark your PCBs at the end of the assignment.

5.1 Version control

Use version control for everything, or else! Learning git is frustrating but is a skill you will not regret.

Your group leader should create a forked copy of the wacky-racers-2020 git project and then add the other group members to the project. This can be done by:

1. Go to `https://eng-git.canterbury.ac.nz/wacky-racers/wacky-racers-2020`
2. Click 'Fork' button. This will create a copy of the main repository for the project.
3. Click on the 'Settings' menu then click the 'Expand' button for 'Sharing and permissions'. Change 'Project Visibility' to 'Private'.
4. Click on the 'Members' menu and add group members as Developers.
5. Using a bash terminal (or other useful shell), enter the command:

   ```
   $ git clone https://eng-git.canterbury.ac.nz/your-userid/wacky-
     racers-2020.git   wacky-racers
   ```

   If you do not want to have to enter your password for every git push/pull operation, you should set up ssh-keys and use:

   ```
   $ git clone git@eng-git.canterbury.ac.nz:your-userid/wacky-racers
     -2020.git wacky-racers
   ```

6. Add a remote URL for the main repository.

   ```
   $ cd wacky-racers
   $ git remote add upstream https://eng-git.canterbury.ac.nz/wacky-
     racers/wacky-racers-2020.git
   ```

   Again if you do not want to manually enter your password, you can use:

   ```
   $ cd wacky-racers
   $ git remote add upstream git@eng-git.canterbury.ac.nz:wacky-
     racers/wacky-racers-2020.git
   ```

   If we add more demo code or tweak the instructions in the main repository, you can get the updated stuff using:

   ```
   $ git pull upstream master
   ```

5.2 Components

1. We recommend that you use components in the ECE Altium library. These are stocked in the SMT lab. For any other components you may require, see Scott Lloyd in the SMT lab.
2. The Wacky Racer batteries are 6-cell NiMH with a three pin connector. To preserve the battery life it is imperative to not draw current when the battery voltage is below 6 V.

5.3 Connectors

1. USB micro or mini connector for debugging
2. 3 pin 0.1" for LED strip (pin 1 5 V, pin 2 signal, pin 3 ground)
3. 2 pin 0.1" for bumper (pin 1 switch, pin 2 ground)
4. 10 pin IDE for serial wire debug

5. 3 pin power connector (pin 1 VBAT, pin 2 GND, pin 3 N.C.)
6. motor connectors (for Racer)
7. connectors for dastardly stuff

5.4 Schematics

1. Have a look at the Altium tutorial on ecewiki.
2. Have a read of the schematic guidelines on ecewiki.
3. Add you and your partner's name and your group number to the title block on your schematic.
4. Save PDF files of your schematics in your source repository. **Note, when debugging your PCBs, we will not help you until you show us your printed schematic**.
5. We bet that you will not have enough test points to clip an oscilloscope probe to. Do not think you can hold the probe tip against an MCU pin. Ensure you give a meaningful name to the test point. A ground test point is essential for an oscilloscope earth clip. Keep this clear of other test points since the clip may short against them. You will probably require at least two ground points.
6. Checking the schematic is the most crucial part of the assignment. If the schematic is wrong then your PCB will be wrong. So, schematics must be thoroughly checked by another person.
7. Consider fall-back options if you have a problem with your PCB.

   The IMU for the Wacky Hat is tiny and we **strongly recommend** that you provide an alternative connector for connecting the following IMU module: MPU-9250 on AliExpress

   Similarly for the Wacky Hat, in case the H-bridge fails, provide two three-pin servo connectors so that external Electronic Speed Controllers (ESC) can be used to drive the motors.
8. It would be useful to have a jumper or two connected to a PIO pin so that you can configure your board. For example, if a jumper is in, use the joystick, otherwise use the IMU.

5.5 PCBs

1. Your four-layer PCBs are going to be manufactured in batches. There is at least a week turnaround time to get the boards made.
2. It is important that you check footprints for parts they you create. We will impose a 10% penalty for each rerun of a PCB, say due to a footprint mistake. Get your partner to check.
3. PCB layouts must be thoroughly checked by another person.
4. A PCB track can blow faster than a fuse. So keep high current tracks fat and short.
5. Clearly mark the positive and negative battery connections on the silk screen.
6. Some of the chips can get hot so thermal considerations are required. Follow the manufacturers' guidelines in the datasheets.
7. The switching regulators can interfere with the radios.

8. Use a design rule check to see if any of the following constraints are violated:
   - Minimum trace width (0.15 mm)
   - Minimum trace clearance (0.15 mm)
   - Minimum via size (0.3 mm hole, 0.6 mm outer diameter)
   - Minimum hole size (0.3 mm)
   - Minimum annular ring (0.1 mm)

   For every violation of one these rules, we will deduct 1% from your final mark.
9. Check the PCB checklist on ecewiki before submission.

5.6 Assembly

1. Finding shorts is extremely frustrating so maximise clearances and test for shorts before populating components.
2. Components can be put through the oven on the reverse side although heavy components may need to be glued.
3. Never assume where pin 1 is on an IC; check the datasheet. 5–10% of groups will get this wrong.

5.7 Software

1. We highly recommend using a personal laptop with Linux installed if possible. A virtual machine running on Windows is acceptable for this. You will need to check instructions on ecewiki for how to install the required toolchain.
2. We do not support embedded systems development on Windows, but it is possible.
3. If you are not using version control for this you are foolish.
4. Inspect MPH's sample code.

5.8 Programming

1. If you are trying to program the SAM4S for the first time and are feeling tired or impatient, then do something else.
2. For the first program, do not use batteries or a USB connection. The ST-Link adapter will provide 3.3 V to the MCU.
3. Detailed instructions can be found at `http://ecewiki.elec.canterbury.ac.nz/mediawiki/index.php/Wacky_racers_software`.

5.9 Debugging

1. Start running small programs (such as the provided demo programs) to test each feature separately.
2. An oscilloscope is your friend.

3. It is possible to use the GDB debugger but you need to know what you are doing, especially with optimised code.
4. Drawing a diagram of what you think is happening is highly recommended. A simple circuit diagram or timing diagram will often help you realise what you have missed and let you fix it without asking for help.

5.10 Possibly asked questions with answers

- *Why use the SAM4s MCU?* For this application most MCUs would suffice, even an 8-bit AVR microcontroller. To level the playing field, I have chosen a MCU most students would not have used before. This is an ARM based MCU made by Atmel I have used this in a number of projects. Indeed we used to teach this chip in ENCE361. There are many other similar MCUs made by different manufacturers such as the STM32 that would just as suitable.
- *Why use a four layer PCB?* Come to lectures to find out!
- *Why use 7.2 V NiMH batteries for the Wacky Racers?* These were a legacy of previous Wacky Racers. They are also safer than lithium batteries with sleep-deprived students.
- *Why can't I program my device using Windows?* Due to various aspects of the Windows ecosystem, setting up a fully functional build environment is more complicated, will often vary from machine to machine, will often break filepaths, and will often cause bizarre compilation errors. Use at your own risk, we will not assist in debugging issues related to Windows.

6 Assistance

- Emails to the lecturers (except of a personal nature) will be quietly ignored.
- Questions anwered on the ENCE461 Learn discussion page will be promptly answered.
- All decisions regarding legality of your dastardly devices are at the whim of Ben Mitchell. You can email Ben at `mailto://ben.mitchell@pg.canterbury.ac.nz` if you wish to keep your ideas secret.
- TAs will be available in the scheduled lab times. Priority will be given to groups assigned to the current lab session. We will only provide assistance to students who have a printed A3 schematic sheet in front of them and have already tried looking up the problem on `https://ecewiki.elec.canterbury.ac.nz`.
- Questions pertaining to Altium and surface mount rework will be answered by Scott Lloyd (SMT Lab technician) and Diego Ramirez (Electronics Lab technician).
- Michael Hayes is really busy pulling all the string behind the scenes and so will only help with gnarly problems referred to by the TAs!